CERTIFICATE OF TRANSMISSION

I hereby certify that this correspondence (along with any paper referred to as being attached or enclosed) is being submitted *via* the USPTO EFS Filing System on the date shown below to **Mail Stop Appeal Brief-Patents**, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Date: January 30, 2009 /Jessica Sexton/
Jessica Sexton

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:

Applicant(s):	Nithyalakshmi Sampathkumar, *et al.*	Examiner:	Nathan Hillery
Serial No:	09/901,368	Art Unit:	2176
Filing Date:	July 9, 2001	Conf. No:	6483

Title: XSLTRANSFORM

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

Applicant submits this brief in connection with an appeal of the above-identified patent application. Payment is being submitted via credit card in connection with all fees due regarding this appeal brief. In the event any additional fees may be due and/or are not covered by the credit card, the Commissioner is authorized to charge such fees to Deposit Account No. 50-1063 [MSFTP296US].

I. Real Party in Interest (37 C.F.R. §41.37(c)(1)(i))

The real party in interest in the present appeal is Microsoft Corporation, the assignee of the present application.

II. Related Appeals and Interferences (37 C.F.R. §41.37(c)(1)(ii))

Appellants, appellants' legal representative, and/or the assignee of the present application are not aware of any appeals or interferences which may be related to, will directly affect, or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. Status of Claims (37 C.F.R. §41.37(c)(1)(iii))

Claims 20-37 have been cancelled. Claims 1-19 and 38 stand rejected by the Examiner. The rejection of claims 1-19 and 38 is being appealed.

IV. Status of Amendments (37 C.F.R. §41.37(c)(1)(iv))

No amendments have been submitted after the Final Office Action.

V. Summary of Claimed Subject Matter (37 C.F.R. §41.37(c)(1)(v))

A. Independent Claim 1

Independent claim 1 relates to a system for transforming XML items. The system comprises memory configured to receive and store a set of input XML items, and a transformer that transforms the input XML items in a first format to a set of transformed XML items in one or more second XML formats. (*E.g.*, see Specification, p. 4 last paragraph through p. 6 first paragraph). Furthermore, the system can comprise an output manager that facilitates selectively pulling or pushing a subset of the one or more input XML items, the subset being less than the set of XML items (*E.g.*, Specification, p. 11, first full paragraph).

B. Independent Claim 19

Independent claim 19 relates to a computer readable medium storing computer executable components of a system for transforming XML items. The system can comprise memory configured to receive and store an input XML item and a transforming component that transforms the input XML item from a first format to a transformed XML item in one or more second XML formats. (*E.g.*, Specification, p. 4 last paragraph through p. 6 first paragraph). Additionally, the system can comprise an output managing component that facilitates selectively pulling or pushing a subset of input XML items, where the subset is less than all of the input SML items (*E.g.*, Specification, p. 11, first full paragraph). Furthermore, the system comprises a compiling component that compiles a style sheet and produces actions that can be employed in processing associated with XML transformation (*E.g.*, see Specification, p. 11, second paragraph). Additionally, the system can comprise an input abstracting component that presents input XML items having different representations to the transformer in a common representation (claims as originally filed) and a node selection abstracting component that dynamically constructs a subset of input XML items from a set of input XML items in response to a query (*E.g.*, see Specification pp. 16-17).

C. <u>Independent Claim 38</u>

Independent claim 38 relates to a system for transforming streaming XML data. The system can comprise a transformer that receives a set of streaming input XML items and transforms the items from a first format to a second format. (*E.g.*, Specification, p. 4 last paragraph through p. 6 first paragraph). The system can comprise memory for storing the input XML items, and an output manager that facilitates pulling or pushing a subset of transformed XML items, the subset being smaller than all of the input items, to provide selective streaming output of transformed XML items. (*E.g.*, see Specification, pp. 27-28 and Figs. 8-9). Additionally, the system comprises a node selection abstractor that dynamically constructs the plurality of input XML items from a set of input XML items in response to a query. (*E.g.*, see Specification pp. 16-17).

D. <u>Dependent Claim 2</u>

Claim 2 relates to a transformer that comprises an action frame stack that holds one or more actions, an event state machine that tracks state associated with transforming XML items and an event processor that receives events generated in processing actions stored in the action frame stack. (*E.g.*, see Fig. 9 and Specification pp. 27-28).

E. <u>Dependent Claim 3-5</u>

Claims 3-5 relates to a compiler that compiles style sheets and produces actions that can be employed by the transformer in processing XML items (*E.g.*, see Fig. 9 and Specification pp. 27-28).

F. <u>Dependent Claim 6-11</u>

Claim 6 relates to an input abstractor that exposes data stored in one or more data stores in a common representation (*E.g.*, see Specification p. 6, first full paragraph and p. 13, second paragraph).

G. <u>Dependent Claim 12-14</u>

Claims 12-14 relate to a node selection abstractor that constructs subsets of input XML items from a set of XML items in response to a query. (*E.g.*, see Specification, p. 16 last paragraph and p. 17 first paragraph).

H. <u>Dependent Claim 15-18</u>

Claims 15-18 relate to an optimized data store that stores XML items in a manner that facilitates minimized processing for construction of subsets of SML items via query. (*E.g.*, see Fig. 5 at 510 and corresponding written description).

VI. Grounds of Rejection to be Reviewed (37 C.F.R. §41.37(c)(1)(vi))

A. Whether claims 1, 3, 4, 5, and 19 are unpatentable under 35 U.S.C. §103(a) over Kuznetsov (US 6,772,413 B2).

B. Whether claim 2 is unpatentable under 35 U.S.C. §103(a) over Kuznetsov (US 6,772,413 B2) in view of Omoigui (US 2003/0126136 A1).

C. Whether claims 6-18 and 38 are unpatentable under 35 U.S.C. §103(a) over Kuznetsov (US 6,772,413 B2) in view of ADO.NET (English Translation).

VII. Argument (37 C.F.R. §41.37(c)(1)(vii))

A. Rejection of Claims 1, 3, 4, 5, and 19 Under 35 U.S.C. §103(a)

Claims 1, 3, 4, 5, and 19 stand rejected as obvious under 35 U.S.C. §103(a) over Kuznetsov (US 6,772,413 B2, hereinafter Kuznetsov).

Reversal of the rejection is respectfully requested for at least the following reasons:

a. Kuznetsov does not disclose or suggest each of the features recited in claims 1, 3, 4, 5 and 19.

b. Kuznetsov does not enable the subject matter employed in rejecting the above claims, and thus is not an enabling reference when utilized in this context.

c. Modifying Kuznetsov to meet the features recited in claims 1 and 19 would destroy the purpose and function of the respective inventions recited by those claims.

Discussion:

a. As a general overview, independent claims 1 and 19 relate to providing a streaming input and streaming output incremental XML transformer that can be employed in push and/or pull model processing. The XML transformer facilitates a user incrementally building the output from XML data so that only a subset of an XML document needs to be loaded into memory to perform a selective transformation. More specifically, independent claims 1 recites: *memory configured to receive and store one or more input XML items; a transformer that transforms the one or more input XML items in a first format to one or more transformed XML items in one or more second XML formats*; *and an output manager that facilitates at least one of* ***selectively pulling and pushing a subset of the one or more input XML*** *items, the subset of the one or more XML items is less than the whole one or more input XML items*.

In general, Kuznetsov discloses transformation of streaming XML items from one format to another format. Kuznetsov is completely silent with regard to selectively pulling or pushing a subset of input XML items, which provides for selective transformation of XML. The subject matter recited in claim 1 provides a specific benefit over and above Kuznetsov, in that because only a subset of input items need be transformed, significant processor and memory (*e.g.*, cache) overhead is saved. As a more specific example, the output manager can selectively pull or push only a fraction of XML items in a large XML document, providing transformation of only those portions of the document that are pertinent to requested information, for instance. None of these features are provided by Kuznetsov, as Kuznetsov mechanically transforms each and every streaming input XML item into a corresponding streaming output item.

The Final Office Action dated May 01, 2008 cites column 13, line 66 – column 14 line 1 of Kuznetsov to show that any number of translators can be implemented simultaneously, such that an entire set (or selected subset) of packets can be translated during runtime. The Final Action contends that this portion of Kuznetsov discloses the output manager recited in claim 1. However, this is based on a misinterpretation of the disclosure of Kuznetsov.

The cited portions of Kuznetsov simply disclose any number of translators that can be implemented simultaneously to transform an entire set, or presumably a selected subset, of packets. No explanation is given as to how the selected subset can be generated, however. Furthermore, this is not the same as selectively pulling or pushing a subset of input XML items. Even if the number of translators provided by Kuznetsov can produce multiple streams of transformed XML items, there is no mention of distinguishing the streams, no mention of identifying one stream from another, and so on, so it is unclear how any selectivity can be employed. Furthermore, as the Final Office Action concedes, Kuznetsov does not disclose the subset of XML items being less than the whole of the XML items. The Final Action suggests that modifying Kuznetsov to meet this missing limitation would be obvious to one of skill in the art. However, the cited art does not disclose any reason or mechanism for doing so, the Final Action simply concludes this from a single reference in Kuznetsov, discussed above, which is not enabled, as discussed below. Thus, it is submitted, that Kuznetsov does not enable the subject matter recited by claim 1.

b. In addition to the foregoing, Kuznetsov does not enable the subject matter employed in rejecting the above claims, and thus is not an enabling reference when utilized in this context. For instance, no explanation is given as to how "any number of translators" can facilitate at least one of selectively pulling and pushing a subset of the transformed XML items, especially when the translators of Kuznetsov transform every input item indiscriminately. Claim 1 obviates this problem by joining a transformer with an output manager. Kuznetsov is wholly silent in this regard, however.

This argument was raised to Examiner Hillery in the telephonic interview of May 24, 2007 (made part of the written record in the Reply to Office Action dated October 31, 2007). However, the argument was rejected because, in the Examiner's view, independent claims 1 and 19 do not recite ***how*** a subset is selected, and that claiming the specifics of how the claimed subset is selected are required in order to overcome Kuznetsov, whether or not Kuznetsov enables the subject matter cited against claim 1. In effect, *the Examiner is stating that the claims must recite subject matter required to enable the cited art*, in order for the requirements of 35. U.S.C. §102 or 35 U.S.C. §103 to be satisfied.

It is respectfully submitted that this statement confuses the requirements for patentability. Specifically, the §102 and §103 requirements of novelty and non-obviousness, with the 35 U.S.C. §112 requirements of enablement and written description. §112 requires the ***specification*** to enable claimed subject matter, the claim itself does not have to be enabling on its face. Thus, if the specification enables selectively pulling or pushing a subset of input XML items, claim 1 meets the requirements of §112. Whether the requirements of §102 or §103 are met are determined based (in part) on whether a prior art reference teaches each feature of a claim (§102) or suggests to one of skill in the art to combine teachings of different references or modify the art to include missing features (§103). However, a prior art reference ***must enable*** subject matter that it is cited as teaching, regardless of whether or not the words descriptive of the subject matter are parroted in one form or another by the reference.

> A prior art reference must be enabling as required for U.S. patents under 35 U.S.C. §112, first paragraph. *Paperless Accounting, Inc. v. Bay Area Rapid Transit Sys.*, 804 F.2d 659, 665, 231 USPQ 649, 653 (Fed. Cir. 1986), *cert. denied*, 480 U.S. 933 (1987). The description must enable a person with ordinary skill in the art not only to comprehend the invention but also to make it. *Paperless Accounting, Inc., v. Bay Area Rapid Transit Sys.*, 804 F.2d at 665, 231 USPQ at 653.

Accordingly, it is submitted that the Examiner is in err in rejecting claims 1 and 19 for at least this additional reason.

While it is true that, as Examine Hillery mentioned in the above telephonic interview, an issued U.S. patent, such as Kuznetsov, used as a prior art reference is presumed to be valid, and that claims are presumed supported by an enabling disclosure (*see Ex parte* Goldgaber, 41 USPQ2d 1172, 1175 (B.P.A.I. 1996) (citing *in re* Lamberti, 545 F.2d 747, 751 n.2, 192 USPQ 278, 281 n.2 (C.C.P.A. 1976), that presumption extends *only* to the claimed subject matter of the U.S. patent. *Kuznetsov does not claim* the portion of the disclosure employed to teach "…an output manager that facilitates at least one of selectively pulling and pushing a subset of the one or more input XML items, the subset of the one or more XML items is less than the whole one or more input XML items". Accordingly, it is submitted that *the presumption of enabling disclosure does not extend* to *unclaimed subject matter*, specifically, "any number of translators can be implemented simultaneously, such that an entire set (or selected subset) of packets can be translated during runtime" (Kuznetsov, column 13 line 66 through column 14 line 1) as cited by the Office Action of October 31, 2007 for instance, to render claims 1 and 19 obvious (Office Action dated October 31, 2007, p. 3, 4th paragraph). As stated above, the concept of transforming a subset of XML items is only mentioned in passing by Kuznetsov, and nowhere recited in any of its claims. Therefore, the cited portion of Kuznetsov is not enabled as applied to above-cited portions of claims 1 and 19, and thus does not teach or suggest the claimed subject matter and therefore cannot render the claimed subject matter obvious.

c. Furthermore, speculation as to how the "any number of translators" could produce a subset of transformed items, absent the output manager, appears to render claim 1 inoperable.

> If a reference is cited that requires some modification in order to meet the claimed invention or requires some modification in order to be properly combined with another reference and such modification destroys the purpose or function of the invention disclosed in the reference, one of ordinary skill in the art would not have found a reason to make the claimed modification. *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984).

Presumably, although neither Kuznetsov nor the Final Action positively state, one or more of the "any number of translators" might be turned off to facilitate translating a selected subset of packets (such a result is speculation, however, it is not disclosed by Kuznetsov), but even that does not read on "***a translator*** … and an output manager that facilitates at least one of selectively pulling and pushing a subset of the transformed XML items" as turning off the translator of claim 1 would result in no items being transformed. Consequently, such a result would render the claimed subject matter unsuited for its intended purpose. Therefore, one of skill in the art would not modify the disclosure of Kuznetsov to arrive at the subject matter recited by claim 1.

Claim 19 recites, in addition to a transforming component and output managing component similar to the transformer and output manager of claim 1, *a compiling component that compiles a style sheet and that produces one or more actions that can be employed by the transforming component in processing associated with transforming the input XML item; an input abstracting component that presents input XML items stored in one or more different representations to the transforming component in a common representation; and a node selection abstracting component that dynamically constructs a subset of input XML items from a set of input XML items, the subset of input XML items are responsive to a query.*

Kuznetsov appears to be wholly silent with regard to at least the node selection abstraction component that dynamically constructs a subset of input XML items from a set of input XML items, the subset of input SML items are responsive to a query. Thus, claim 19 enables a query to identify a portion of XML items to be transformed and input to the transforming component, to allow for selective transformation or selective output of transformed XML items. Kuznetsov is silent regarding at least this feature of claim 19. Moreover, because Kuznetsov discloses only automatic transformation of streaming input, providing a corresponding transformed XML item for each input XML item, the capabilities provided by claim 19 are wholly unattainable by Kuznetsov. For example, Kuznetsov states that "According to the invention [of Kuznetsov] … [w]hen fed an input data stream, the data translator generates an output data stream by executing the native object code…" (Kuznetsov, column 5, lines 4-5 and 14-16), without mention of selectively pulling or pushing portions of the input data stream or output data stream. Furthermore, when an XML protocol is employed, "an XML stream translator can be completely replaced or augmented by an optimized translator operated according to the present invention…. This illustrative embodiment comprises an optimized contiguous memory algorithm, the performance of which approaches that of a memory-to-memory copy utility…" (Kuznetsov, column 5, lines 42-47 and 50-54). It is submitted that a memory-to-memory copy utility is typically refers to a brute-force copy or transformation mechanism from a first memory to another, without any selective intelligence. Accordingly, it is submitted that Kuznetsov provides no motivation for one of skill in the art to modify its disclosure to arrive at the missing features recited by claim 19.

In view of the foregoing, it is believed that Kuznetsov does not teach or suggest each and every aspect of independent claims 1 and 19 (and claims 3, 4, and 5 that depend there from). Therefore, it is respectfully requested that this rejection be overturned.

B. Rejection of Claim 2 Under 35 U.S.C. §103(a)

Claim 2 stands rejected as obvious under 35 U.S.C. §103(a) over Kuznetsov in view of Omoigui (US 2003/0126136 A1, hereinafter Omoigui).

Reversal of this rejection is requested for at least the following reason: Kuznetsov and Omoigui, either alone or in combination, do not teach or suggest each and every aspect set forth in the subject claims.

> "Under 35 U.S.C. 103 where the examiner has relied on the teachings of several references, the test is whether or not the references viewed individually and collectively would have suggested the claimed invention to the person possessing ordinary skill in the art. It is to be noted, however, that citing references which merely indicated that isolated elements and/or features recited in the claims are known is not a sufficient basis for concluding that the combination of claimed elements would have been obvious. That is to say, there should be something in the prior art or a convincing line of reasoning in the answer suggesting the desirability of combining the references in such a manner as to arrive at the claimed invention... [I]t would not have been obvious to modify [the prior art] ... without using [the patent application's] claims as a guide. It is to be noted that simplicity and hindsight are not proper criteria for resolving the issue of obviousness." *Ex parte Hiyamizu*, 10 USPQ2d 1393 (BPAI 1988).

Claim 2 recites: the system of claim 1, the transformer comprises an action frame stack that holds one or more actions, an event state machine that tracks state associated with transforming the one or more XML items and an event processor that receives events generated in processing the one or more actions stored in the action frame stack. As is discussed below, the Omoigui reference bears not the slightest relevance to these features recited in claim 2, and therefore fails to cure the deficiencies of Kuznetsov.

In general, Omoigui discloses an integrated framework and medium for knowledge retrieval, management, delivery and presentation. A server component is responsible for adding and maintaining domain-specific semantic information and a second server component hosts semantic and other knowledge for use by the first server component. The server components work together to provide context and time-sensitive semantic information retrieval services to clients operating a presentation platform via a communication medium. Overall, Omoigui has little relevance to selective XML transformation provided by the subject application and, generally speaking, recited in the subject claims.

However, the above Final Office Action cites a particular portion of Omoigui to cure the deficiencies of Kuznetsov in regard to claim 2, under a heading of "Security" which reads as follows: "The preferred embodiment of the Information Nervous System provides support for all aspects of security: authentication, authorization, auditing, data privacy, data integrity, availability, and non-repudiation. This is accomplished by employing standards such as WS-Security, which provides a platform for security with XML Web Service applications. Security is preferably handled at the protocol layer via security standards in the XML Web Service protocol stack. This includes encrypting method calls from clients (semantic browsers) to servers (Agencies), support for digital signatures, authenticating the calling user before granting access to an Agency's Semantic Network and XML Web Service method, *etc.*" (Omoigui, paragraph 0367). It is submitted, however, that the XML Web Service protocol stack discussed by Omoigui, which deals primarily with providing encryption for web applications, is related to the "action frame stack" recited in claim 2 by nothing more than the five letters that form the word 'stack'. Accordingly, Omoigui does not disclose or even suggest the action frame stack, state manager and event processor recited by claim 2.

An "XML Web Service protocol stack" is not equivalent or suggestive of an action frame stack that holds one or more actions. An action stack, as recited in claim 2, can be considered analogous (although not necessarily equivalent) to a queue that can store actions while other actions are being processed (*e.g.*, by the event processor, also recited in claim 2). It is unclear from Omoigui or from the Final Action exactly what the XML Web Service protocol stack of Omoigui is. However, it is submitted that a Web services protocol stack in general is merely a collection of protocols to define standards for electronic or digital communication. For instance, below is cited at least one article to support this notion. According to Wikipedia, a Web services protocol stack is:

> "a collection of computer networking protocols that are used to define, locate, implement and make Web services interact with each other. The Web service protocol stack mainly comprises four areas: a (Service) Transport Protocol responsible for transporting messages between network applications and includes protocols such as HTTP, SMTP, FTP … an (XML) Messaging Protocol response for encoding messages in a common XML format so that they can be understood at either end of a network connection … a (Service) Description Protocol used for describing the public interface to a specific web service … and a (Service) Discovery Protocol that centralizes services into a common registry such that network web services can publish their location and description, and makes it easy to discover what services are available on the network…"

It is submitted that the XML Web Service protocol stack of Omoigui is merely a collection of computer networking protocols pertinent to providing "method calls from client [device]s, support for digital signatures, authenticating the calling user before granting access to an Agency's Semantic Network and XML Web Service methods" (Omoigui, paragraph 0367). It is manifest, however, that a set of protocols do not equate to, teach, or suggest *an action frame stack that holds one or more actions*, as recited by claim 2. Protocols define syntax, organization, structure and the like for software code. They do not, in and of themselves, perform activities such as holding one or more actions (*e.g.*, in queue for a processor), as does the action frame stack of claim 2.

In addition to the foregoing, Omoigui does not disclose or suggest *an event state machine that tracks state associated with transforming the one or more XML items* and further does not disclose or suggest *an event processor that receives events generated in processing the one or more actions stored in the action frame stack.* Particularly, Omoigui does not include either of these features anywhere within its disclosure. Neither does Omoigui imply that these items are included within the XML Web Service protocol stack. Such contention appears for the first time in the Final Action, wholly unsupported by the cited art. The Final Action therefore fails to show how the cited art discloses each and every feature of claim 2 to one of ordinary skill in the art, or provides a suggestion to modify the cited art to arrive at missing features recited by the claim. Rather, the Final Action misinterprets the art with respect to one feature of claim 2 (the action frame stack) and simply ignores two other features (the event state machine and event processor) of the claim.

In view of the foregoing, it is submitted that neither Kuznetsov nor Omoigui, alone or in combination, disclose, or suggest each and every feature of claim 2. Therefore, it is respectfully requested that this rejection be overruled.

C. Rejection of Claims 6-18 and 38 Under 35 U.S.C. §103(a)

Claims 6-18 and 38 stand rejected as obvious under 35 U.S.C. §103(a) over Kuznetsov as applied to claim 1 above and further in view of ADO.NET (English Translation, hereinafter ADO.NET).

Reversal of this rejection is requested for at least the following reasons:

a. ADO.NET is not prior art with respect to the cited claims

b. ADO.NET and Kuznetsov do not teach or suggest each and every aspect of the above claims; and

c. ADO.NET is not an enabling reference with respect to the subject matter it is cited against.

Discussion:

a. ADO.NET appears to be an Internet document, presumably retrieved from an online database, cited by the Final Action dated May 1, 2008 to render obvious portions of claims 6-18. However, ADO.NET provides no publication date. Further, the Final Action provides no date as to when the reference was retrieved. It is submitted, therefore, that ADO.NET cannot be relied upon as prior art under 35 U.S.C. §102(a) or (b), or by extension 35 U.S.C. §103(a).

> Prior art disclosures on the Internet or on an on-line database are considered to be publicly available as of the date the item was publicly posted. Absent evidence of the date that the disclosure was publicly posted, *if the publication itself does not include* ***a***

> ***publication* date (or *retrieval* date), it cannot be relied upon as prior art** under 35 U.S.C. 102(a) or (b). MPEP §2128.

The Final Action appears to rely on a date printed on the cover sheet of ADO.NET, namely, March 6 – 8, 2001, in determining when ADO.NET was "described in a printed publication in this or a foreign country, before the invention…by the applicant for patent" (35. U.S.C. §102). However, this date is insufficient proof that ADO.NET was publicly available prior to the filing date of the subject application. MPEP §2128 is explicit: an Internet reference must have a publication date or retrieval date prior to the date of invention (July 09, 2001). ADO.NET does not provide a publication date, a copyright date, or any other evidence that suggests the March 6-8, 2001 date refers to a date of publication. It is submitted that any desired date can be printed on an internet document; the printing of such date, however, does not imply that the document was available to the public (*e.g.*, hosted by an Internet site and properly catalogued by a search engine) on that date, as required by the MPEP. As a hypothetical example, a date of March 6-8, 2015 can easily be printed on any internet document, just as March 6-8, 2001 appears on the cover of ADO.NET. However, clearly March 6-8, 2015 could not correspond to a date of public availability for the hypothetical example, because such point in time has not yet come to pass. It is submitted that an arbitrary date on the cover of an Internet reference, without more evidence, is insufficient proof of a date of publication, as required by MPEP §2128.

Further to the above, it is noted that the cited date (March 6-8, 2001) appears on the cover of the reference in conjunction with a name of a city in Denmark (Copenhagen), under a title ADO.NET. It is submitted, that this date is probably a date on which a conference on the ADO.NET subject matter was held at Copenhagen, Denmark. However, it is further submitted that this hypothesis provides no evidentiary weight to show that the following pages, which the Final Action cites against the subject claims, were published or otherwise publicly available on this date.

Based on the foregoing, the only date that can be relied on for the ADO.NET reference is a date that the examiner personally downloaded it from the Internet, or Office personnel retrieved the document. Further to this end, it is noted that another date does appear on the cover of ADO.NET, specifically March 2005, in conjunction with UNITED STATES PATENT AND TRADEMARK OFFICE Washington, D.C. It is further submitted that this is likely the retrieval date of the ADO.NET reference by Office personnel. Because the Final Action has failed to show any publication date for ADO.NET, this reference should be accorded a date no earlier than its retrieval date by the USPTO, March 2005. Because this retrieval date is after the filing date of the subject application, ADO.NET does not qualify as prior art with respect to the above claims.

b. In addition to the foregoing, the Final Action fails to establish a *prima facie* case of obviousness because neither Kuznetsov nor ADO.NET teach or suggest all the elements of claim 1, from which claims 6-18 depend, or the additional elements recited by claims 6, 8, 9, 10, 12, 15-18, and newly added claim 38. Specifically, with regard to claim 1, Kuznetsov and ADO.NET do not disclose *a transformer* that transforms one or more input XML items in a first format to one or more transformed XML items in one or more second XML formats*;* and *an output manager* that facilitates at least one of selectively pulling and pushing a subset of the input XML items, the subset of the one or more XML items is less than the whole one or more input XML items. As discussed above, he Final Action recites column 13 lines 66 through column 14 line 1 of Kuznetsov to render obvious "an output manager that facilitates at least one of selectively pulling and pushing a subset of the transformed XML items".

The cited portions of Kuznetsov simply teach any number of translators that can be implemented simultaneously to transform an entire set of packets; no explanation is given as to how a selected subset of packets can be translated. As discussed above, Kuznetsov does not enable the subject matter recited by claim 1. Therefore, "any number of ... translators can be implemented simultaneously, such that an entire set (or selected subset) of packets can be translated" (Kuznetsov, column 13 line 66 through column 14 line 1) does not teach or suggest the subject matter of claim 1. In addition, ADO.NET is silent with respect to this element of claims 6-18, and therefore does not cure the deficiencies of Kuznetsov.

In addition to the foregoing, independent claim 38 recites *a transformer* that receives a plurality of input XML items and transforms the input XML items from a first format to one or more second XML formats; *an output manager* that facilitates at least one of selectively pulling and pushing a subset of the transformed XML items, the subset having fewer XML items than the plurality of input XML items, to provide a selective streaming output, the selective streaming output is provided to a memory; and *a node selection abstractor* that dynamically constructs the plurality of input XML items from a set of input XML items, the plurality of input XML items are responsive to a query. It is submitted that neither Kuznetsov nor ADO.NET teach or suggest the above-cited aspects of claim 38. Specifically, neither Kuznetsov nor ADO.NET teach or suggest a transformer in conjunction with an output manager that facilitates at least one of selectively pulling and pushing a subset of transformed XML items. Further, neither Kuznetsov nor ADO.NET teach or suggest a node selection abstractor that dynamically constructs the plurality of input XML items … responsive to a query. Accordingly, it is submitted that claim 38 recites patentable subject matter in view of the cited art.

Furthermore, with regard to claim 6, the Examiner cites ADO.NET to teach "an XpathNavigator is created to abstract data from the xml data set via an XpathNodeIterator by employing a loop", and contends that this portion of ADO.NET renders obvious: *an input abstractor that exposes data stored in the one or more data stores in a common representation* (*e.g.*, *see* Office Action dated October 31, 2007, p. 8, 5th paragraph). This would require that all abstraction of data employing a loop exposes the abstracted data in a common representation. Nothing in the reference supports this result, and the Office Action does not suggest it either. Therefore, the cited portion of ADO.NET is simply insufficient to meet the subject matter recited in claim 6.

With regard to claim 8, the Final Action recites page 19 of ADO.NET to teach an XpathNavigator is created to abstract data from the xml data set. The Final Action claims that this subject matter renders obvious "the input abstractor exposes the data stored in the one or more data stores as a data model and infoset". This statement is logically incorrect, however, as it would require all data abstracted from an xml data set to be exposed in one or more data stores as a data model and infoset. This is not the case. Consequently, the cited portion of ADO.NET is insufficient to meet the claimed subject matter recited by claim 8.

With regard to claim 9, the Final Action recites page 19 of ADO.NET to teach an XpathNavigator is created to abstract data from the xml data set and sends the data to an XSLT. The Final Action claims that this subject matter teaches "the input abstractor provides a cursor model over data stored in a data store to facilitate presenting a stream of nodes to the transformer", as recited by claim 9. It is not apparent how abstracting data and sending said data to an XSLT meets "providing a cursor model over data stored in a data store to facilitate presenting a stream of nodes to a transformer". This would require all abstraction of data to necessarily include “providing a cursor model over data stored in a data store to facilitate presenting a stream of nodes”. The ADO.NET reference certainly does not disclose this proposition, and no other reference cited in the Final Action suggests this result either. Consequently, ADO.NET is insufficient to meet the subject matter of claim 9.

With regard to claim 10, the Final Action recites page 19 of ADO.NET to teach an XpathNavigator is created to abstract data from the xml data set. The Final Action contends that this cited portion of ADO.NET discloses “the input abstractor provides a virtual node that can be employed to traverse the stream of nodes” recited by claim 10. However, this result also requires all data abstracted from an xml data set to “provide a virtual node that can be employed to traverse the stream of nodes.” ADO.NET does not disclose this proposition, and no other reference cited in the Office Action suggests this result either. Consequently, the cited portion of ADO.NET is insufficient to meet the subject matter of claim 9.

With regard to claim 12, the Office Action recites page 19 of ADO.NET to teach “SQL is used to query xml items and store them to an XML data set and that each node in the xml dataset is visited by employing an XpathNodeIterator. The Office Action contends that this cited portion of ADO.NET discloses “a node selection abstractor that dynamically constructs a subset of input XML items from a set of input XML items, the subset of input XML items are responsive to a query”. An SQL query does not necessarily dynamically construct a subset of input XML items from a set of input XML items. Such a result would be necessary or at least implicit for the cited portion of ADO.NET to teach or suggest the subject matter of claim 12.

In addition, nothing in ADO.NET or in Kuznetsov provides motivation to combine an SQL query with constructing a subset of input XML items from a set of input XML items in conjunction with a transformer that transforms one or more input XML items in a first format to one or more transformed XML items in one or more second XML formats, as recited by claim 12.

> “Under 35 U.S.C. 103 where the examiner has relied on the teachings of several references, the test is whether or not the references viewed individually and collectively would have suggested the claimed invention to the person possessing ordinary skill in the art. It is to be noted, however, that citing references which merely indicated that isolated elements and/or features recited in the claims are known is not a sufficient basis for

> concluding that the combination of claimed elements would have been obvious. That is to say, there should be something in the prior art or a convincing line of reasoning in the answer suggesting the desirability of combining the references in such a manner as to arrive at the claimed invention... [I]t would not have been obvious to modify [the prior art] ... without using [the patent application's] claims as a guide. It is to be noted that simplicity and hindsight are not proper criteria for resolving the issue of obviousness." *Ex parte Hiyamizu*, 10 USPQ2d 1393 (BPAI 1988).

The Final Action contends that doing so would have been obvious because it provides a benefit of "explicit implementation of XPath via source code." (*e.g.*, *see* Office Action dated October 31, 2007, p. 11, 6th paragraph and p. 12 1st paragraph.) However, constructing a benefit in hindsight does not meet the requirement of "something in the prior art or a convincing line of reasoning in the answer suggesting the desirability of combining the references in such a manner as to arrive at the claimed invention…." In this case, the Examiner has provided nothing to show that the above line of reasoning was available to one of skill in the art prior to the filing date of the subject application; in other words, the Examiner provides nothing to show that the reasoning was in the prior art and not constructed purely from hindsight. No reference cited by the Office Action acknowledges such a benefit, nor does the Office Action provide any credence to the fact that such a benefit would have been considered by one of skill in the art prior to the filing date of the subject application. Consequently, the Final Action fails to provide a sufficient *prima facie* case of obviousness with respect to claim 12.

With regard to claim 15, the Final Action recites pages 18-19 of ADO.NET to teach "that SQL is used to query xml items and store them to an XML data set". The Final Action further contends that this subject matter discloses "an optimized data store that stores one or more XML items in a manner that facilitates minimizing processing associated with constructing the subset of input XML items via a query." However, this contention requires that all SQL queries of xml items stored to an XML data set facilitates minimizing processing associated with constructing the subset of input XML items via a query. Nothing in ADO.NET proposes this requirement, and the Office Action gives no additional grounds to suggest this conclusion. In addition, as stated above with regard to claim 12, the Final Action provides insufficient motivation to combine Kuznetsov and ADO.NET, as nothing in either reference and nothing else provided in the Office Action suggests that one of skill in the art would be motivated to combine the references by "the benefit of explicit implementation of XPath via source code". This conclusion seems to be manufactured wholly at first instance by the Office Action; the cited art does not support it. Consequently, it is submitted that the cited art neither teaches nor suggests each feature recited by claim 12, nor does there exist a sufficient motivation to combine such references as required by MPEP §706.02.

With regard to claims 16 and 18 the Final Action recites pages 18-19 of ADO.NET to teach that an "XPath document is created and used to store and manipulate the xml data set". The Office Action contends that this cited portion discloses "the optimized data store stores data in a data representation format that facilitates optimizing an XPath query". Similar to analogous conclusions cited with respect to other claims, stated above, this contention is logically flawed in that it requires that every creation of an XPath document that stores and manipulates an xml data set to necessarily include a data representation format that facilitates optimizing an XPath query. This result would logically negate any optimization of the XPath query, because if all queries involved the same data representation that optimizes the query, none would be optimal over another. Consequently, ADO.NET is insufficient to teach the subject matter recited by claim 16.

c. Finally, it is noted that because ADO.NET does not provide a comprehensible description of its activities with respect to interaction of depicted components, it is improper for the Office Action to characterize the illustration as anything other than a depiction of disparate components connected by directional arrows that leaves open the door for mere speculation and conjecture. It is further submitted that one of ordinary skill in the art would not be able to rely on the ADO.NET disclosure to understand the subject matter that the Office Action claims is disclosed by the reference. Thus, the reference is not enabling for the purported teachings asserted by the Office Action.

D. Conclusion

For at least the above reasons, the claims currently under consideration are believed to be patentable over the cited references. Accordingly, it is respectfully requested that the rejections of claims 1-19 and 38 be reversed.

If any additional fees are due in connection with this document, the Commissioner is authorized to charge those fees to Deposit Account No. 50-1063 [MSFTP296US].

Respectfully submitted,
AMIN, TUROCY & CALVIN, LLP

/Himanshu S. Amin/
Himanshu S. Amin
Reg. No. 40,894

AMIN, TUROCY & CALVIN, LLP
57TH Floor, Key Tower
127 Public Square
Cleveland, Ohio 44114
Telephone (216) 696-8730
Facsimile (216) 696-8731

VIII. Claims Appendix (37 C.F.R. §41.37(c)(1)(viii))

1. A system for transforming XML items, the system comprising:

a memory configured to receive and store one or more input XML items;

a transformer that transforms the one or more input XML items in a first format to one or more transformed XML items in one or more second XML formats; and

an output manager that facilitates at least one of selectively pulling and pushing a subset of the one or more input XML items, the subset of the one or more XML items is less than the whole one or more input XML items.

2. The system of claim 1, the transformer comprises an action frame stack that holds one or more actions, an event state machine that tracks state associated with transforming the one or more input XML items and an event processor that receives events generated in processing the one or more actions stored in the action frame stack.

3. The system of claim 1, further comprising a compiler that compiles one or more style sheets and produces one or more actions that can be employed by the transformer in processing associated with transforming the one or more input XML items.

4. The system of claim 3, the compiler resolves one or more external references in the one or more style sheets.

5. The system of claim 4, the input XML items are input from one or more data stores.

6. The system of claim 5, further comprising an input abstracter that exposes data stored in the one or more data stores in a common representation.

7. The system of claim 6, the input abstractor abstracts a reference to a node within an XPath document.

8. The system of claim 7, the input abstractor exposes the data stored in the one or more data stores as a data model and infoset.

9. The system of claim 8, the input abstractor provides a cursor model over data stored in a data store to facilitate presenting a stream of nodes to the transformer.

10. The system of claim 9, the input abstractor provides a virtual node that can be employed to traverse the stream of nodes.

11. The system of claim 10, the input abstractor is an XPathNavigator.

12. The system of claim 6, further comprising a node selection abstractor that dynamically constructs a subset of input XML items from a set of input XML items, the subset of input XML items are responsive to a query.

13. The system of claim 12, the node selection abstractor facilitates navigating the subset of input XML items.

14. The system of claim 13, the node selection abstractor is an XPathNodeIterator.

15. The system of claim 12, further comprising an optimized data store that stores one or more XML items in a manner that facilitates minimizing processing associated with constructing the subset of input XML items *via* a query.

16. The system of claim 15, the optimized data store stores data in a data representation format that facilitates optimizing an XPath query.

17. The system of claim 16, the data representation format comprises expanded XML entities, deleted XML declarations and DOM model data converted to XPath model data.

18. The system of claim 17, the optimized data store is an XPathDocument.

19. A computer readable medium storing computer executable components of a system for transforming XML items, the system comprising:

a memory configured to receive and store an input XML item;

a transforming component that transforms the input XML item from a first format to a transformed XML item in one or more second XML formats;

an output managing component that facilitates at least one of selectively pulling and pushing a subset of the input XML item, the subset of the input XML item is less than the whole input XML item;

a compiling component that compiles a style sheet and that produces one or more actions that can be employed by the transforming component in processing associated with transforming the input XML item;

an input abstracting component that presents input XML items stored in one or more different representations to the transforming component in a common representation; and

a node selection abstracting component that dynamically constructs a subset of input XML items from a set of input XML items, the subset of input XML items are responsive to a query.

20 - 37 (Cancelled).

38. A system for transforming streaming XML data, the system comprising:

a transformer that receives a stream of a plurality of input XML items and transforms the input XML items from a first XML format to one or more second XML formats;

memory configured to receive and store the stream of the plurality of input XML items;

an output manager that facilitates at least one of selectively pulling and pushing a subset of the transformed XML items, the subset having fewer XML items than the plurality of input XML items, to provide a selective streaming output, the selective streaming output is provided to the memory; and

a node selection abstractor that dynamically constructs the plurality of input XML items from a set of input XML items, the plurality of input XML items are responsive to a query.

IX. Evidence Appendix (37 C.F.R. §41.37(c)(1)(ix))

None.

X. Related Proceedings Appendix (37 C.F.R. §41.37(c)(1)(x))

None.